TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Model autonomního vozu sledujícího čáru SC3

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Petr Vošoust**
Vedoucí práce: Ing. Jan Koprnický, Ph.D.

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Autonomous Line Following Car Model SC3

Bachelor thesis

Study programme: B2646 – Information technologies
Study branch: 1802R007 – Information technologies

Author: **Petr Vošoust**
Supervisor: Ing. Jan Koprnický, Ph.D.

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr Vošoust**
Osobní číslo:	**M11000137**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Model autonomního vozu sledujícího čáru SC3**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s elektromechanickými částmi podvozku a řídicí jednotkou.
2. Zprovozněte hardwarovou část systému.
3. Navrhněte algoritmus řízení a vytvořte odpovídající software pro řídicí jednotku modelu vozu.
4. Funkční model otestujte na závodní dráze.

Rozsah grafických prací:	**dle potřeby dokumentace**
Rozsah pracovní zprávy:	**30–40 stran**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **The Freescale Cup Knowledge Center. FREESCALE. The Freescale Cup Knowledge Center [online]. 2012 [cit. 2012-10-04]. Dostupné z: https://community.freescale.com/docs/DOC-1284**

[2] **Ďaďo, S.; Kreidel, M.: Senzory a měřicí obvody. Praha: ČVUT, druhé vydání, 1999, ISBN 80-01-02057-6.**

[3] **FREESCALE. MPC560xB Controller Board User's Guide. MPC560XBMCBUG, Rev. 0, 08/2012. 2012.**

Vedoucí bakalářské práce:	**Ing. Jan Koprnický, Ph.D.** Ústav mechatroniky a technické informatiky
Datum zadání bakalářské práce:	**10. října 2014**
Termín odevzdání bakalářské práce:	**15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2014

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Abstrakt

Tato bakalářská práce se zabývá návrhem inteligentního algoritmu, který vývojovému kitu v podobě vozítka zaručí projetí testovací trati v nejrychlejším možném čase a bez porušení oficiálních pravidel pro soutež EMEA Freescale Cup 2014. Vozítko pomocí lineární kamery snímá tmavou čáru na světlém podkladě a je vybaveno vývojovými deskami plošných spojů a sadou elektromotorů, které umožní pohon i natočení kol.

Klíčová slova: MCU, EMEA Freescale Cup 2014, algoritmus řízení, TRK vývojový kit, sledování čáry

Abstract

This Bachelor´s thesis describes the design of an intelligent algorithm for vehicle like development kit. Vehicle have to pass the test track in the fastest time as possible and without breaking the official rules for contest EMEA Freescale Cup 2014. Vehicle uses a linear camera to monitor a dark line on a light background and is equipped with development boards of printed circuit. And with a set of electric motors to allow the drive and steering.

Keywords: MCU, EMEA Freescale Cup 2014, driving algorithm, TRK development kit, line monitoring

Poděkování

Předně bych chtěl poděkovat doktoru Koprnickému, za odborný dohled i úpravu „nečeských“ výrazů ve více srozumitelné. Za rady i jiné pohledy na problematiku i formu této BP. Kolegům Markovi Křivanovi, Lukáši Vlčkovi a Davidu Václavkovi za konzultace. Své přítelkyni Petře Staňkové za fotodokumentaci. Svým rodičům za podporu u studia. Vedoucímu a majiteli obchodu s elektronikou v ul. 5. května (S+K Elektro) za vstřícnost a rady ohledně nákupu elektroniky. Doktoru Novákovi za smysluplné a odborné rady a závěrem docentu Satrapovi za vynikající šablonu pro kvalifikační práce v LaTeXu.

Obsah

Seznam obrázků

Seznam zkratek

2D	2-Dimension – dvourozměrný
AO	Analog output – čtení z posuvného registru
CPU	Central Processor Unit – standartní počítačový procesor
CK	Clock – Singnál od krystalu
CW	CodeWarrior, název vývojového prostředí
DEMO	Demonstration – předvedení
DPS	Deska plošných spojů
EMEA	Europe Middle East Africa
EU	Evropská unie
GND	Ground – uzemění, zem
I/O	Input/Output – vstup/výstup
IDE	Integrated Development Environment – vývojové prostředí
MCU	MicroController Unit – mikrokontrolér
PWM	Pulse Width Modulation – pulzně šířková modulace
SE	Second Edition – druhé vydání
SI	Serial Input – Data sériové linky, čtení připraveno
USB	Universal Serial Bus

Úvod

Tato bakalářská prace se zabývá nejdříve motivační částí a to, co autora vedlo ke zvolení tohoto tématu. Dále jsou pak popsány v teoretické části jednotlivé komponenty konstrukce, pak elektroniky a nakonec využitého softwaru, který sloužil při sestrojení algoritmu. V další (praktické) části zabývající se řešením jsou vyčteny jednotlivé body zadání. Uvedeno bude seznámení se s vývojovým kitem a jeho možnostmi. Poté zprovoznění hardwaru a tvorba algoritmu, přičemž budou popsána úskalí, kdy probíhalo zdržení celé práce a také principy fungování na mapě myšlenek.

Popsány budou základní funkce, které tvořily jádro funkčního inteligentního algoritmu, který dokázal vozítko řídit a sledovat přitom čáru.

1 Motivace a přístup k řešení

1.1 Soutěž

Motivací bylo splnit zadání tak, abych se mohl účastnit soutěže EMEA 2014 pořádané společností Freescale Cup, soutěž samořízených autíček sledujících čáru. Navíc mě přitahují ryze technické obory jako je například robotika a kybernetika a svoji budoucí profesi bych si představoval právě v tomto odvětví. První krok tímto směrem by mohlo být splnění podmínek pro tuto soutěž.

Cílem práce je projet „neznámou“ trať v co možná nejkratším čase a hlavně bez kolize.

1.1.1 Trať

Základem trati je světlý (bílý) vystouplý podklad (50 cm široký) a na něm vyznačena spojitá tmavá (černá) čára 1 palec široká. Díly resp. úseky trati jsou předem dány avšak složení trati nikoli. Pro mé testování byl použit jednoduchý okruh tvořen z těchto dílů. Díly zahrnují:

- rovinku,
- zatáčku,
- startovní čáru.

1.1.2 EMEA 2014

Soutěž EMEA 2014 Freescale cup je tradiční soutěž pořádáná pro technické univerzity po celém světě. Studenti se snaží vyřešit zadání a neporušit žádnou ze série striktních podmínek.

Tato mezinárodní soutěž od prvního konání v roce 2012 stále přitahuje nové studenty, ale i univerzity po celém světě a Česká republika není výjimkou. Pro zajímavost, soutěže se v roce 2014 účastní 48 univerzit z 16 různých zemí a celkový počet týmů přesahuje číslo 140.

Pravidla se každým rokem upravují a jsou stále více technicky zaměřená pravě pro tento druh studentů. Znění pravidel pro rok 2014 je dostupné zde [5] na webových stránkách komunity Freescale.

2 Teoretická část a rešerše

Od vedoucího bakalářské práce jsem na začátku semestru obdržel balení s vývojovým kitem a všemi díly. Balení obsahovalo:

- Základní šasi vozítka.
- Přední náprava se servomotorem.
- Zadní náprava s dvěma elektromotory
- Řídicí vývojovou desku s MCU.
- Výkonovou desku s H-můstky na řízení elektromotorů.
- Desku s vektorovou kamerou.
- Baterii.
- Propojovací kabely.

Jednotlivé díly budou popsány a ilustrovány dále.

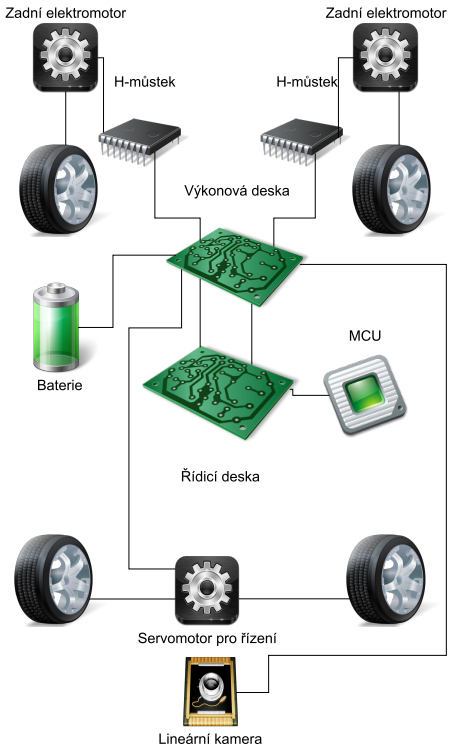
2.1 Konstrukce

Pod tímto termínem se rozumí mechanické vybavení vývojového kitu. Budou zde popsány jednotlivé části v několika podkapitolách.

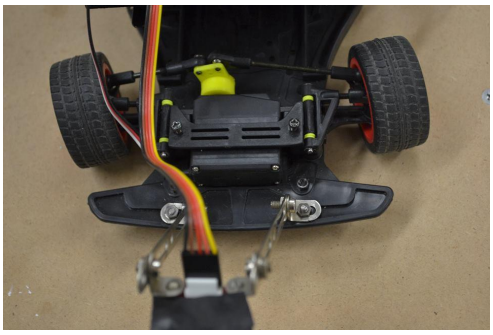
Vývojový kit je tvořen plastovým podvozkem, který má dvojici předních a zadních kol, která jsou tvořena gumou na plastových ráfcích. Úplně vpředu je připevněná kamera na stojánku pomocí dílu ze stavebnice Merkur. Za ní je situován servomotor a vzadu dva nezávislé elektromotorky – motory budou popsány v následující podkapitole 2.1.1. Uprostřed podvozku je položená 7,2V baterie (viz ilustrace 3.3a), která slouží jako zdroj elektrické energie pro napájení dvou DPS i motorů. Blokové schéma na obrázku 2.1 ilustruje propojení celého systému.

2.1.1 Řízení

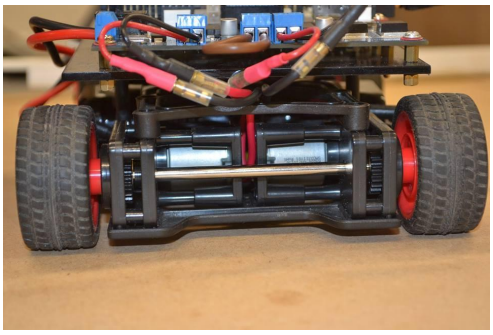
Vývojový kit disponuje třemi elektromotory nápájenými stejnosměrným proudem (7,2 V). Jeden je v přední části a tento servomotor se stará o natáčení předních kol (viz obr. 2.2a). Dále pak má dvojici elekromotorů v zadní části (viz obr. 2.2b), přičemž každý pohání jedno z kol.



Obrázek 2.1: Schéma konstrukce vývojového kitu



(a) Servomotor v přední části



(b) Elektromotory v zadní části

Obrázek 2.2: Elektromotory

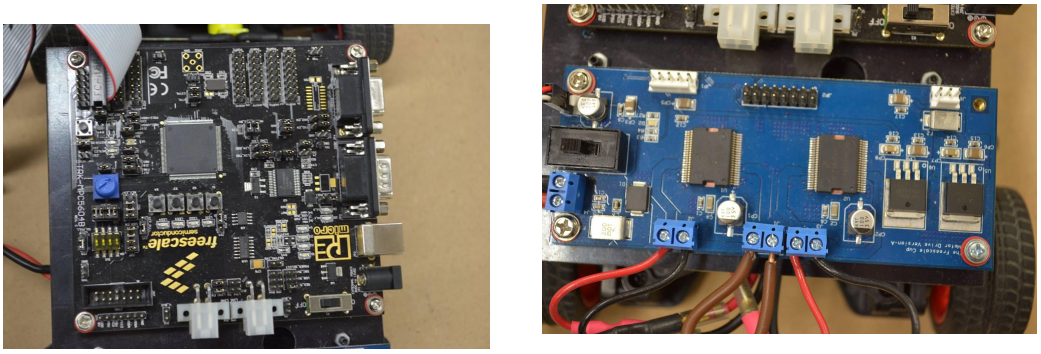
2.1.2 Kamera

Vektorová (řádková) CMOS kamera, dodaná v balení, je připevněna na úplném začátku šasi). Je připevněna pomocí kovového stožárku a předsunuta před vozítko, aby mohla snímat dráhu. Snímá pouze řádkový vektor o 128 hodnotách, přičemž se jedná o černobílé rozlišení s 8-bitovou hloubkou. Je tvořena čipem TSL1401CL od firmy TAOS a 128 fotodiodami v jednořádkovém uspořádání. Je řízena třemi signály. CK signál je tik od časovače, který je tvořen krystalem o frekvenci 8 MHz. Dále pak SI (serial input), který nám zahájí čtení kamery a AO (analog output), který

naopak zajistí čtení analogových hodnot z posuvného registru, kde jsou uloženy hodnoty z jednotlivých fotodiod. Další informace o kameře jsou dostupné na webových stránkách komunity Freescale, které jsou zmíněny v použité literatuře [12].

2.2 Hardware

Tato kapitola popisuje eletronické vybavení. Budou zde popsány 2 desky plošných spojů od společností Freescale Semiconductors a P&E Microcomputer Systems (ilustrace obou desek na obr. 2.3). Pro přehled si je rozdělme na desku „hlavní“ a „podpůrnou“. Obě DPS jsou umístěny na plastové desce držící na sloupích z šasi. Ta, která je blíže kameře (umístěna v přední části), bude označována jako řídicí deska (viz kapitola 2.2.1), ta blíže k zadním elektromotorům bude popisována jako deska výkonová (viz kapitola 2.2.2). Obě budou podrobněji popsány v následujících podkapitolách.



(a) Hlavní (řídicí) deska plošných spojů

(b) Podpůrná (výkonová) deska plošných spojů

Obrázek 2.3: Obě desky plošných spojů

2.2.1 Hlavní deska

Hlavní deska (černé barvy, obr. 2.3a), Qorivva TRK-MPC5604B se stará o veškeré řízení a výpočty. Nachází se na ní mikrokontrolér obr. 2.4 (mikropočítač v kapitole 2.2.1), který představuje tlukoucí srdce celého systému. Veškeré vstupy a výstupy jsou řízeny také přes tuto desku a je zde několik možností komunikace s okolními komponentami. Podle obrázku 2.3a je vidět, že deska obsahuje spoustu vstupních i výstupních pinů a portů (viz kapitola o portech 2.2.1).

Hlavní deska také obsahuje paměti typu SRAM (32 kB) a FLASH (512 kB), což umožňuje nahrát program přímo do paměti, či jej uložit na FLASH uložiště a program spuštet opakovaně i po odpojení od zdroje napájení (jak USB tak baterie).

Mimo to hlavní deska disponuje i tlačítkem na reset MCU, což je zejména výhodné při nefunčnosti algoritmu, aby nedošlo k mechanickému poškození částí kitu. Také 4 další programovatelná tlačítka a 4 zelené LED diody pro různou (nastavitelnou) signalizaci během vývoje.

Deska je posázena také nastavovatelnými piny (pomoci jumperů) a nejdůležitější z nich, z hlediska vývoje, je 6pinový úsek J1 (viz [15] nebo na přiloženém CD – příloha A), kde si lze zvolit typ napájení (z baterie či externí).

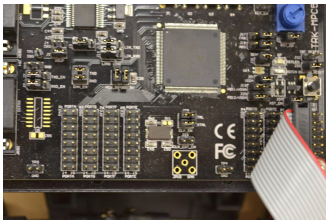
MCU MPC5604B

Mikrokontrolér z rodiny Qorivva je jádrem celého kitu. Nejedná se o procesor (CPU) nýbrž o mikropočítač, který procesor již obsahuje. Je to jednočipový 32bitový počítáč, který může samostně fungovat, zejména i proto, že má vlastní FLASH i RAM paměť a programovatelné vstupy a výstupy jako jsou například A/D a D/A převodníky.

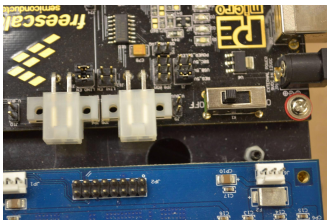
Vstupy a výstupy

Z portů je nejvíce patrný na první pohled port USB typu A. Ten slouží ke komunikaci s počítačem, ve kterém je vyvíjen algoritmus řízení. Hned vedle jsou také známé sběrnice a to CAN a RS232 (obr. 2.4a), které se dnes standardně využívají jako komunikační sběrnice. V horní části je dvojice sběrnic typu LIN (obr. 2.4c). V opačné části desky jsou situovány programovatelné porty (obr. 2.4b) PORT# (kde # je písmeno A – F). Skrze nastavení registrů lze pinům určit funkci jako vstupní i jako výstupní. PORTB (viz schéma řídicí desky [15]) je již předprogramován v DEMO aplikaci na komunikaci s podpůrnou deskou, kde je zejména využita pulzně šírková modulace (PWM).

(a) USB, RS232 a CAN



(b) I/O Porty a MCU



(c) Sběrnice LIN

Obrázek 2.4: Konektory na hlavní desce

2.2.2 Podpůrná deska

Deska modré barvy se stará o napájení celého systému a komunikaci s řádkovou kamerou. Nachází se zde 2 H-můstky, které samostatně pomocí PWM řídí zadní motory. H-můstky se obecně využívají díky nastavitelnosti výstupu, kdy je možné přepólovat elektromotor. Avšak v tomto konkrétním případě reverzace není možná, jelikož je výstup pro zpětné otáčení motoru přípájen na GND. Výstup do předního servomotoru je také na této desce, avšak tento elektromotor není řízený přes H-můstek, ale přímo přes vodič. Dvojice konektorů po obvodu slouží k propojení vodičů s motory a s baterií pod plastovou deskou, nesoucí obě DPS.

2.3 Software

Tato část se bude zabývat programovým vybavením vývojového kitu a jaký software a jeho prostředky budou při realizaci využity.

2.3.1 Užitý software

Společnost Freescale vyvinula vývojové prostředí pro programovaní svých výrobků. To je obecně známo pod zkratkou CW či jeho plným názvem CodeWarrior. V tomto IDE bude tedy algoritmus na řízení vyvíjen. Společně s ním je duležitý software od společnosti P&E Microcomputer Systems s názvem Terminal. Jedná se o terminál/konzoli pro komunikaci uživatele s MCU za běhu programu.

Pro grafické znázornění a tvorbu grafů bude využit matematický software Matlab od společnosti MathWorks.

Pro „rychlé“ naprogramování a zobrazení výstupních dat logiky, co si nelze ihned představit, bude užito IDE NetBeans od společnosti Oracle.

2.3.2 Vývojové prostředí

CodeWarrior je dodáván v různých verzích a je stažitelný na webových stránkách společnosti Freescale Semiconductors. Cena, či dostupnost, se právě odvíjí od požadované verze. Jelikož nám na vývoj algoritmu stačí jen omezené množství paměti a tento algoritmus nenaplňuje svou podstatou komerční využití, tak je naše dostupná verze zdarma.

Bylo potřeba vzít v úvahu, že společnost Freescale Semiconductors vydává mnoho rodin MCU a vývojových desek. Pro náš typ (a obecně pro Qorivva MPC56xx a MPC55xx) je doporučena verze 2.10 SE, která je opět dostupná na webových stránkách (viz [11]).

2.3.3 Programovací jazyk

U vývojových desek našeho typu se často využívá jazyku C nebo přímo Assembler). IDE CW využívá především rychlé a efektivní programovací jazyky Assembler, C a C++. Přičemž pro tuto desku byl využit právě programovací jazyk C, který je hierarchicky nad Assemblerem, avšak stále velice efektivní pro tento typ úlohy.

Výše bylo zmíněno IDE NetBeans, to využivá programovací jazyk JAVA. Nikterak nesouvisí využití tohoto jazyka i prostředí pro vývoj řídicího algoritmu, avšak bude užitečným pomocníkem při psaní funkcí algoritmu mimo rozdělanou práci.

2.3.4 DEMO program

Na stránkách komunity zabývající se vývojem je dostupný předpřipravený DEMO program (viz literatura [14]), který ukazuje, jak pracovat s komponentami vývojového kitu. Je zde předvedeno, jak pracovat s kamerou, jak se hodnoty ukládají, jak

pracovat s jednotlivými elektromotory, nastavení PWM apod. Ve velice jednoduchém zpracování se nováček rychle a efektivně seznámí s logikou, která je užita při vývoji na této konkrétní DPS, než kdyby pročetl datasheet se všemi možnostmi kitu. Podrobněji bude tato část algoritmu popsána v kapitole 3.4.1.

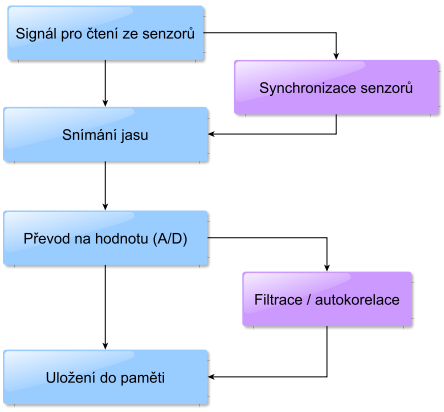
2.4 Sledování čáry

Sledování čáry lze provádět několika způsoby. Nejznámější a nejvíce užívaná je detekce čáry pomocí odrazu světla. V podstatě se jedná o vyhodnocení reflexe povrchu, který kamera (či čidlo) snímá. Výsledkem pak může být pole hodnot jasu, které můžou mít podobu jen typu Boolean (true a false), ale i různou bitovou hloubku znázorňující pokles odrazu. Důležité, při této možnosti, je hlavně osvětlení, protože čára musí být (i její podklad musí být!) na všech úsecích tratě co nejpodobnější. Dalším faktorem je dostatečné odlišení (kontrast) mezi podkladem a čárou. V tomto případě je neideálnější černá a bílá barva jakožto dvě naprosto opačné skutečnosti. Samozřejmě je i možností čáru poupravit tak, aby její povrch odrážel více či méně světla, avšak údržba trati by byla časem překážkou.

Pokud kamera snímá nějaký obraz (ať už řádkový či 2D), záleží i na zpracování. V našem případě je klasifikace celkem jednoduchá a to zjistit, kde a co je čára. Neujžívanější jsou dvě metody vyhodnocení signálu: zvolení prahu (viz ilustrace 2.6) či korelace podle vzoru (nástin principu v ilustraci 2.7). Obě mají své výhody a nevýhody. Pro více informací následujte do použité literatury (položka [2]).

2.4.1 Sledování lineraní kamerou

Blokové schéma na obrázku 2.5 naznačuje, jak takové sledování čáry obecně funguje. Hlavní linie (modrá barva v barevné verzi) je v případě potřeby doplněna o mezikroky (fialová v barevné verzi).



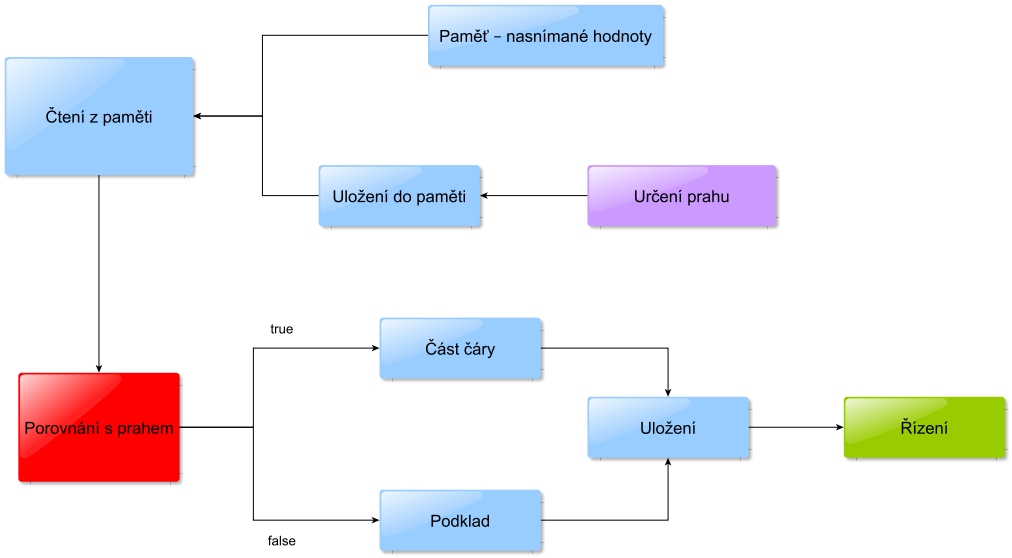
Obrázek 2.5: Schéma sledování čáry lineární kamerou

Prvotně záleží z kolika čidel je kamera sestavena, protože je potřeba signál sesynchronizovat s hodinovým taktem. Pak se snímá samotný signál představující odraz

čteného řádku. Analogový signál se převede na diskrétní a odečtou se jednotlivé hodnoty. Ty se v případě nutnosti filtrují nebo se provede autokorelace. Následně se pak data uloží do paměti či uložiště v žádané podobě či struktuře. Pak následuje zpracování dat a vyhodnocení. Pro více informací o použité řádkové kameře a jejím použití vyhledejte v literatuře položku [12].

2.4.2 Zpracování pomocí prahovací funkce

Obecně se jedná o nastavení limitu, který jasně určí, kdy daná hodnota je čára a kdy ne.



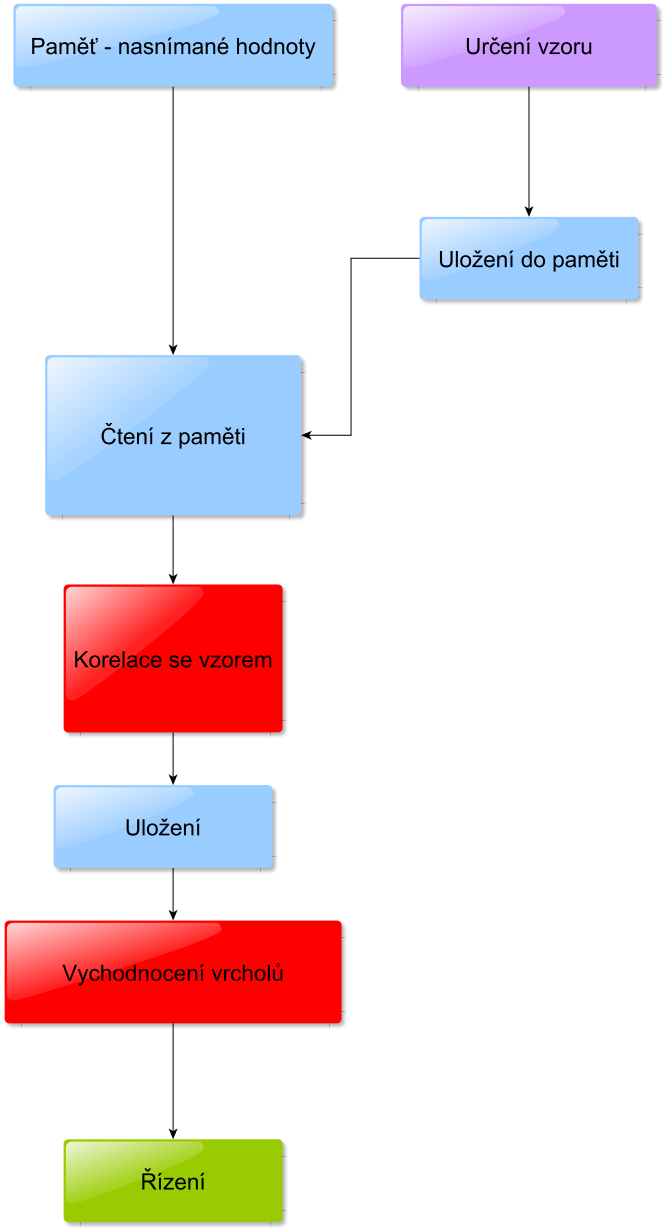
Obrázek 2.6: Schéma principu použití prahovací funkce

Práh lze určit různými způsoby – např. předem stanoveným odhadem či průměrem. Ten i s hodnotami, které jsou již uložené v paměti, se použije k posouzení o lokaci čáry. Každá hodnota se porovná s prahem a v případě, že je nižší/vyšší (rovnost záleží na navrhovateli algoritmu), pak rozhodne o její klasifikaci, jestli je to část čáry či jen okolní podklad.

Výhody tohoto algoritmu jsou jeho jednoduchost a rychlost, avšak je náchylnější na chyby. Práh je fixní a změna obecné odrazovosti (např. snížením osvětlení) je pak fatální. Samozřejmě je možné prahování jakkoli modifikovat a přidávat ošetřující podmínky, ale tím roste jeho složitost a ztrácí tak své výhody uvedné výše.

2.4.3 Zpracování pomocí korelace

Korelace je obecně užívaný standard při zpracování signálů. Ta je obecně daleko náročnější na výpočet, avšak oproti prahování se umí vypořádat se změnou osvětlení. Je ale kritické vhodně zvolit kvalitní vzor, podle kterého se korelace provádí.



Obrázek 2.7: Schéma principu použití korelace

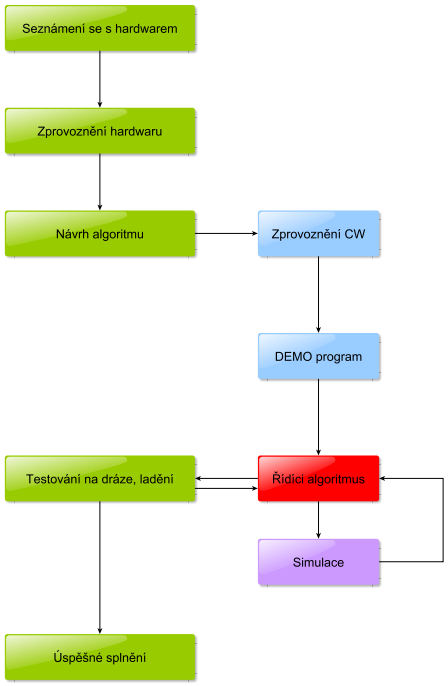
Algoritmus načte z paměti nadefinovaný vzor a uložené hodnoty z kamery. Provede korelaci podle vzorce posouvání indexů. Výsledné hodnoty pak uloží do paměti a z nich se pak vyhodnocuje podobnost oproti vzoru. Pro informace ohledně korelace digitálních a spojitých signálů naleznete v kapitole se seznamem s použitou literaturou (položka [9]).

3 Realizace

V této kapitole bude popsán postup řešení. Od jednotlivých bodů zadání přes mapu myšlenek (obr. 3.1) až k samotné realizaci všech kroků postupu. Některé cesty však nebyly úplně vhodné, avšak i nesprávná cesta posune vykonavatele úlohy blíže k cíli.

3.1 Metodika postupu

Postup při realizaci této bakalářské práce byl přesně v souladu se zadáním.

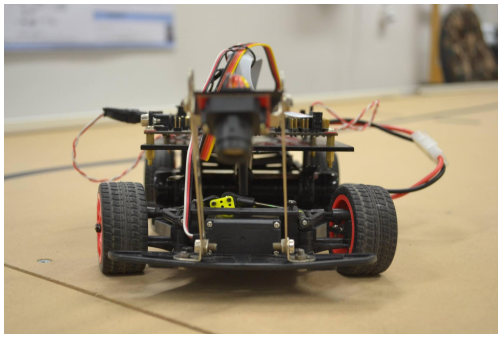


Obrázek 3.1: Metoda postupu

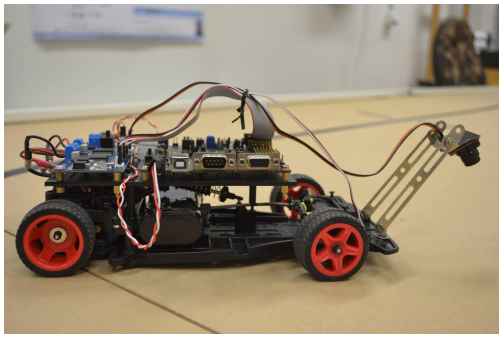
Nejdříve bylo potřeba celý kit zkontrolovat, jestli obsahuje všechny části a také jestli jsou nepoškozené, aby výměna proběhla co nejrychleji. Po kontrole kitu přišla na řadu kontrola hardwaru; tedy obou DPS. To zahrnovalo i pročtení datasheetů dostupných na webových stránkách. Pak realizace algoritmu samotného a průběžné a finální testování.

3.2 Seznámení se s hardwarem

Byla provedena kontrola celého vývojového kitu (ilustrace kitu 3.2). Nacházel se v něm model s již přimontovanou plastovou destičkou k šasi. I obě DPS byly již přichyceny šroubky tak, aby nedošlo ke zktratu. Kamera byla připevněna v přední části pomocí stavebnice Merkur na předsunutém sloupku konstrukce. Baterie umístěna vespod byla přichycena pomocí dvou posuvných plastových stahovacích pásků, byla tak zabezpečena oproti posunu při náklonu či neopatrém zacházení. Motory byly pevně zasazeny do zadní části *chassis* a spojeny s otočnými zařízeními pro pohon kol. Servo v přední části mělo již připevněno své otočné rameno k poloosám natáčejícím obě kola.



(a) Vozítko – pohled na frontální část



(b) Vozítko – pohled ze strany

Obrázek 3.2: Vývojový kit

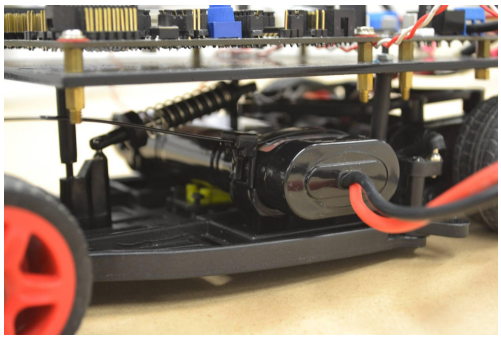
3.3 Zprovoznění hardwaru

Obě desky byly již propojené, avšak bylo nutné vše zkontrolovat, aby nedošlo k poškození. K úspěšné kontrole posloužil zdokumentovaný návod, dostupný zde [7]. PORTB (kapitola 2.2.1) byl obsazen vodičem propojujícím obě desky. Kamera s řízením servomotoru byla připojena v k podpůrné desce (viz kapitola 2.2.2).

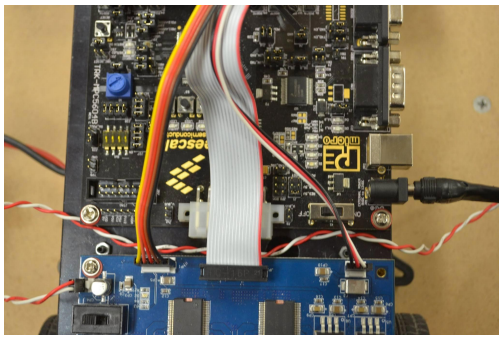
Zadní motory však správně zapojeny nebyly. Tento nedostatek byl odhalen, když byl prvně zkompilován a spuštěn DEMO program (popsaný v kapitolách 2.3.4 a 4). Jeden z motorů měl totiž prohozené vodiče a otáčel se opačným směrem. Avšak náprava byla velice jednoduchá, stačilo pouze propojit správné vodiče, aby nedocházelo k negativnímu přepólování.

Dalším nedostatkem se po bližším prozkoumání jevil konektor, který sdílel napájení mezi podpůrnou deskou a deskou hlavní. Byl realizován pouze navinutím vodičů ke kontaktům patice a oblepen izolepou. Aby nedošlo k žádnému nevhodnému kontaktu, ať už mezi vodiči samotnými či ostaním hardwarem, byly konektory připájeny a oblepeny elektrikářskou páskou. Jumper v oblasti J1 byl nastaven na externí zdroj – napájení z USB.

Vývojový kit byl ozkoušen a plně zprovozněn až při spuštení demoprogramu.



(a) Zdroj elektrické energie na trati – 7,2V baterie



(b) Propojení hlavní desky a vedlejší s okolními komponentami

Obrázek 3.3: Zprovozněný hardware

3.4 Přípravy a instalace

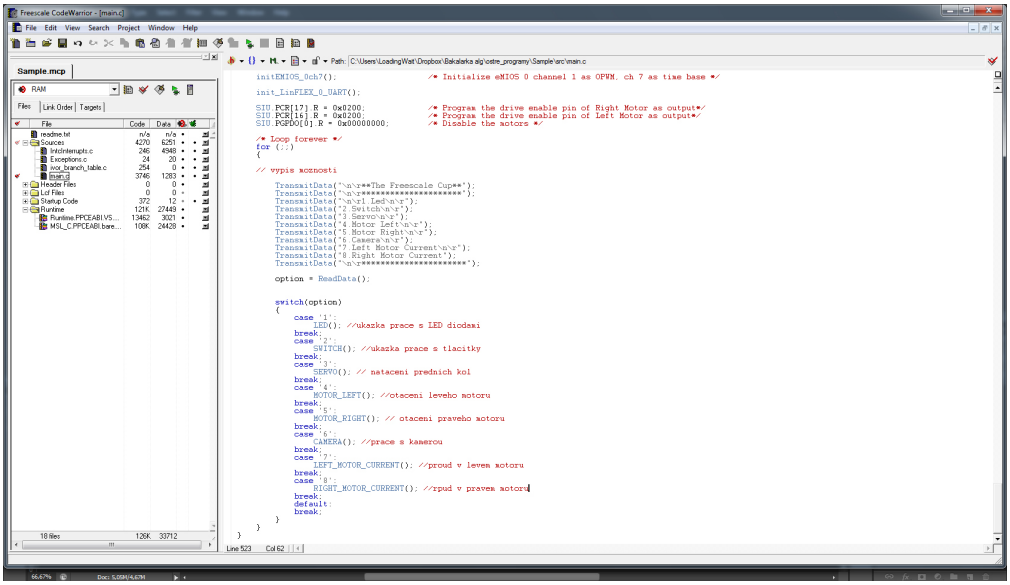
Prvotně bylo nutné nainstalovat verzi v2.10 Codewarrior. Pak i podpůrný software – zejména tedy Terminal, který byl součástí celého balíčku pro vývoj od společnosti P&E Microcomputers. Stáhnutí a zprovoznění DEMO programu, ukazující funcionalitu celého kitu, bylo klíčovým a dalším krokem (viz [14]).

3.4.1 Ukázkové funkce DEMO programu

Program, nebo spíše projekt (viz ilustrace 3.4), který byl stažen a spuštěn v IDE CW, obsahoval plno užitečných řádků kódu. Samozřejmě kromě klíčových definic všech vstupů a výstupů zde byly i ukázany funkce demonstrující jednotlivé možnosti kitu i vývojového prostředí samotného.

Program ukazuje, jak lze řešit vstupy/výstupy pomocí klávesnice v Terminalu, kde se odehrává hlavní běh programu, jak připravit *delay* a další. Dle obrázku také DEMO program ukazuje, jak využít jednotlivé funkce, které byly definovány výše v programu.

Pro více informací lze demoprogram stáhnout, viz literatura [14].



Obrázek 3.4: Design vývojové prostředí a ukázka délky a elegantní jednoduchosti hlavní smyčky DEMO programu

Funkce DEMO programu ve smyčce

1. První funkcí je ovládání zelených LED diod. Ty jsou na hlavní desce celkem 4 a lze je využít pro zpětnovazebnou signalizaci. Program ukazuje, jak postupně jednu za druhou rozsvěcet. Jsou ovládány skrze registr, do kterého se maskováním zapisují hexadecimální hodnoty, díky čemuž pak lze dojít k závěru, že deska je ovládána opačnou logikou (tzv. aktivní v nule).

2. Druhá funkce naznačuje práci s tlačítky, které jsou rovněž na hlavní desce ve stejném počtu jako LED diody – dokonce seřazeny hned ve stejné oblasti (viz obr. 3.5). Také jsou řízeny pomocí registrů.

3. Další funkcí je ukázka práce se servomotorem. Zápis určitých hodnot v rozmezí 1250–1500 do řídicího registru zaručuje natočení kol v požadované mírě.

4. Funkce s pořadovými čísly 4 a 5 ukazují, jak roztočit zadní kola, přičemž je z porovnání obou funkcí patrné, jak motory povolit (ať už jeden nebo dedukcí oba) a zase zakázat. Také je možné zvolit směr otáčení díky H-můstkům na podpůrné desce, avšak tato varianta vývojového kitu tuto možnost nemá (jak je uvedeno v kapitole 2.2.2). Rychlost otáčení (hodnota napětí), ta však definována není, tu lze řídit nastavením PWM (viz kapitola 3.4.1).

5. Poslední důležitou funkcí na ukázku je ovládání kamery. Avšak nejdůležitější částí je uložení hodnot a to do podoby jednorozměrného pole, se kterým se bude v průběhu realizace pracovat a bude se na něj odkazováno v následujících kapitolách.

Obrázek 3.5: Uživatelské LED diody a tlačítka na desce

Užitečné informace k DEMO programu

Celé programování lze uskutečnit při volání funkcí z nekonečného cyklu umístěného v souboru **main.c**. Veškeré funkce lze sekvenčně volat při běhu hlavního cyklu.

Ovládání rychlosti otáčení zadních motorů se provádí skrze nastavení pulzně šířkové modulace. Není potřeba nastavovat generování (o to se postará DEMO program sám), avšak lze měnit opakování se vygenerování jednoho impulzu a tím tak nastavovat délku střídy (konstatně 50 %) a následně i rychlosti vozidla. Vše, co je potřeba nastavit, je registr kanálu 23 (uveden v [6]), který se právě stará o periodičnost signálu PWM. Pro více informací lze pročíst dostupný manuál vývojové desky (viz literatura [7]), který generování popisuje.

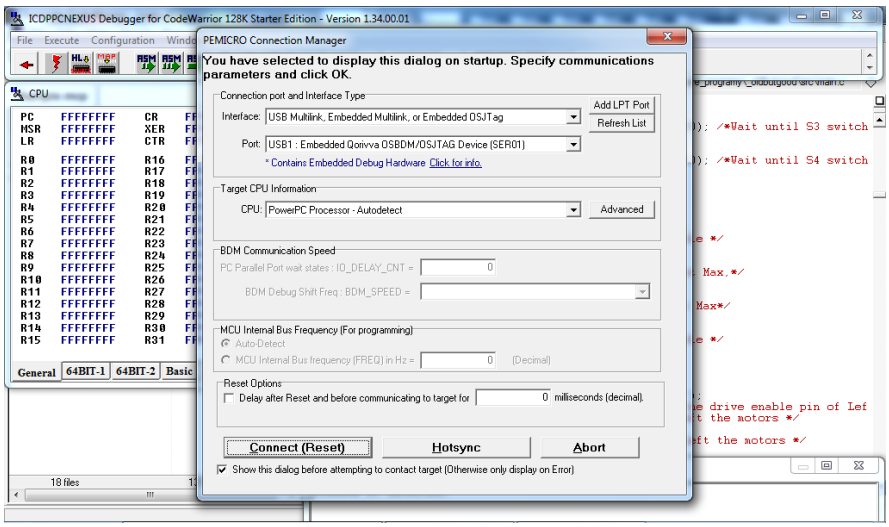
3.5 Zpracování algoritmu

Náš algoritmus, případně předprogramovaný DEMO program, se v prostředí CW zkompiluje a následně je započat proces nahrání do vývojového kitu. Jsou dvě možnosti, jak kód uložit přes USB port.

1. V prvním případě je možné nahrát program pouze do pamětí typu SRAM, tím se proces zrychlí a při odpojení od napájení se volatilní paměti typu RAM po vybití kondenzátorů (prakticky ihned) vymažou.

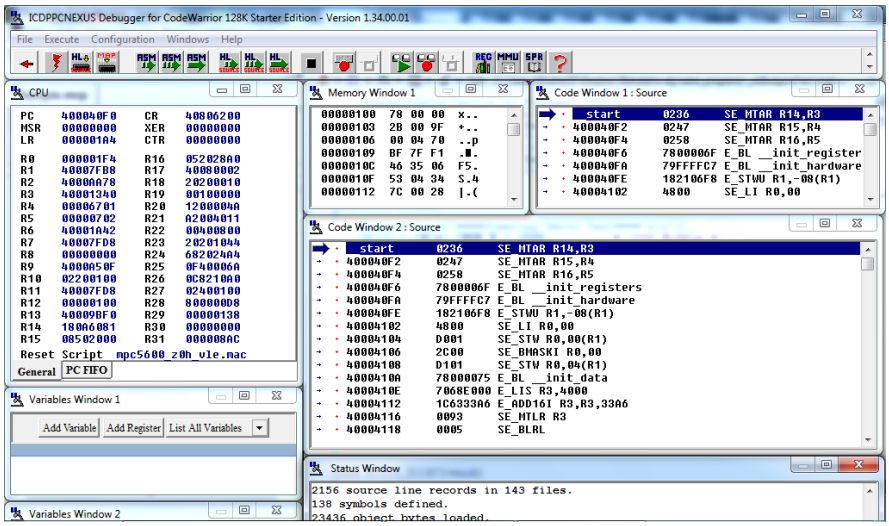
2. Na druhou stranu lze využít i paměti typu FLASH. Proces trvá o něco déle, ale program lze kdykoli při zapnutí a vypnutí kitu zase vyvolat.

Po kompilaci a vytvoření kódu se spustí debugger a začne nahrávání. Je potřeba ještě říci debuggeru, o kterou desku se jedná, ale ve většině případů si ji sám detekuje a jen oznámí, že tu či jinou DPS našel na určitém portu (viz dialogové okno v ilustraci 3.6). Po potvrzení začne debugger kód nahrávat do pamětí a registrů



Obrázek 3.6: Detekce a synchronizace mezi DPS a PC

vozítka. Během procesu je vidět, jak se registry naplňují hodnotami, a jak se vypisují instrukce do jednotlivých řádků představující adresy paměti. Po nahrání, kdy se ukazatel zastaví na první adrese s instrukcí start, je možno spustit program (viz ilustrace s okny 3.7). Možností je spustit program s manuálním krokováním, s krokováním časovače, či v realném čase. Ta je vzhledem ke složitosti a délce algoritmu nejrozumnější.

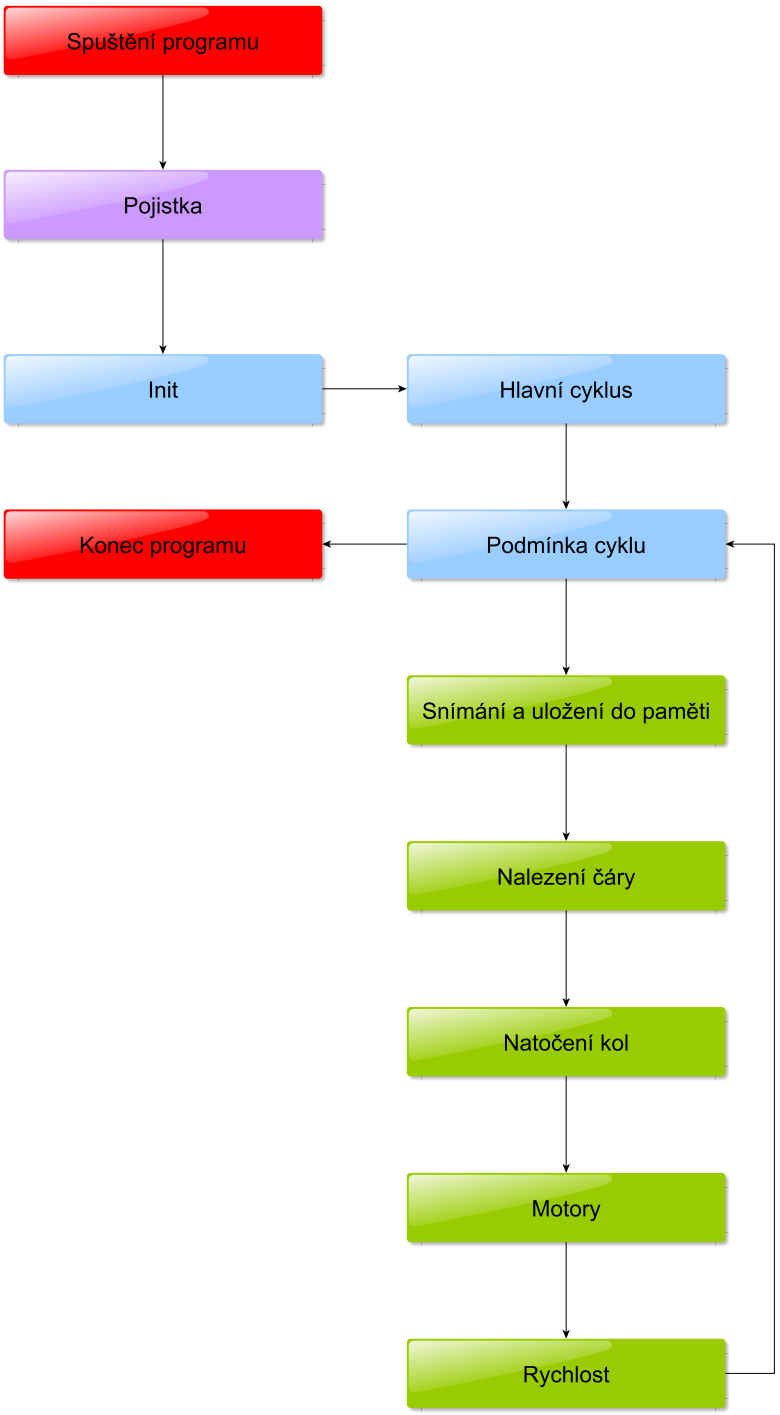


Obrázek 3.7: Nahrávání instrukcí do paměti kitu

Po odkliknutí se okamžitě spustí sekvence příkazů vygenerovaných kompilátorem.

3.6 Návrh algoritmu

Mapa myšlenek, která vznikla po prostudování možností kitu a samotného ukázkového progamu, byla vzorem pro vývoj a princip algoritmu.



Obrázek 3.8: Mapa myšlenek pro návrh algoritmu

Podle ilustrace 3.8, celý program začíná pojistkou, kdy je povolen běh programu od uživatele. Pokud přistoupíme k vykonávání programu, tak se provede inicializace proměných a následně se dostaneme do hlavní smyčky programu.

Kamera nasnímá řádek, který se vyhodnotí a uloží do paměti. Vyhodnotí se hodnoty a nelezne se čára. Kola se natočí směrem ke středu čáry a povolí se oba zadní motory, přičemž se podle natočení upraví rychlost. Na konci se zkontroluje podmínka běhu cyklu a děj se buď opakuje, či je program ukončen.

3.6.1 Pojistka

Program se ihned po nahrání do vozidla začne vykonávat. Proto bylo nutné programu ve vykonávání zamezit. Vznikla funkce **void Ready()**, která program podržela v čekací fázi. Bylo při ní využito i ukázkového programu DEMO, kde se ihned využily ukázkové funkce pro stisknutí tlačítka. Dokud tlačítko není zmáčknuto, program dále nepokračuje. Další užitečnou funkci z DEMO programu bylo ovládání LED diod a zpoždení (tzv. *delay*). Zelená dioda při nahrání programu signalizuje, které tlačítko má být stisknuto, díky tomu, že jsou LED diody na desce paralelně vyrovnány s tlačítky, jak je vidět na obrázku 3.5.

Poté se spustí zpoždění na cca 3 s a pak se přejde na hlavní běh programu ve smyčce.

V prvotní fázi byla smyčka realizována stisknutím mezerníku, protože s kitem bylo pracováno převážně mimo dráhu a na konzoli terminálu.

3.6.2 Init

Při inicializaci dojde k nastavení všech důležitých proměných pro běh hlavní smyčky. Jedna z nich je například **int Cycles**, která se stará o počet běhů smyčky.

3.6.3 Snímání kamery

Algoritmus převezme hodnoty přicházející z čidel kamery (uložené v posuvném registru) a přes A/D převodník je postupně ukládá do statického pole typu *integer*. Uloží zde celkem 128 (každou do samostatného prvku pole) v rozpětí 0–255. O to se stará funkce **void CAMERA()** z DEMO programu. Klíčové však je ukládání do pole a proto byla funkce upravena. Do pole se totiž ukládají hodnoty, které jsou již upravené. Pro zjednodušení a lepší přehlednost vnikla funkce **int cameraBool(int value)**, která převede hodnotu na základě nastaveného prahu na 1 nebo 0. Hodnoty jsou opět typu integer, neboť programovací jazyk C nedisponuje datovým typem boolean. Tuto drobnost nahradil rozhodovací logikou, že true je vše, co není 0. Navíc při hledání čáry se taková jednotnost hodí a významně zjednoduší a zpřehlední algoritmus.

3.6.4 Hledání čáry

Funkce **void seekLine()** se stará o nalezení čáry a nastavení indexu pro natočení kol. Index je hodnota typu *integer* a nabývá hodnot 0–127, kde hodnota je ekvivalentem pozice středu čáry v poli. Zpravidla odladěný (a předem vyzkoušený v NetBeans) algoritmus hledá, kde se v poli nachází největší počet jedniček a index v poli, který leží uprostřed této řady jedniček je pak index, co slouží pro natočení kol. Tato globálně viditelná proměnná **int line_i** je ke konci algoritmu ještě ošetřena o vyjímečné stavy – například vyjetí z trati, přejetí startovní čáry a křížení tratí.

3.6.5 Natočení kol

O natočení kol se dle DEMO programu stará registr, do kterého se ukládají hodnoty mezi 1250 a 1750. Čili z toho vyplývá, že 500 hodnot slouží k natočení autíčka o 90° zleva doprava (45° na obě strany do středu).

Když nyní máme obraz a v něm informaci, kde se nachází čára, a dokonce i index o kolik je posunuta, můžeme natočit kola. Algoritmus, který toto řeší, je vcelku jednoduchý. Prakticky se snaží, aby se index čáry blížil polovině velikosti pole, čili hodnotě 64. A proto, když je čára na krajích pole, natáčí kola vždy ke středu. Když je index uprostřed našeho pole, nedojde k natočení kol a vozítko pokračuje rovně.

3.6.6 Řízení motorů

V této fázi se původně, na základě indexu čáry, pouštěl levý či pravý motor samostatně. Avšak simulace diferenciálu běžného vozu byla tak slabá a neúčinná, že se v této části programu pouze zapíše do registru motorů hodnota, aby se oba točily koly vpřed.

3.6.7 Volba rychlosti

Anglicky pojmenovaná funkce **void speed()** se stará o volbu 3 rychlostí. V zásadě vyhodnocuje odchylku od stavů, kdy jsou kola natočena rovně a pak podle stupně odchylky pouští do motorů 3 ruzné rychlosti v podobě napětí řízeného střídou PWM. Pokud je odchylka malá až žádná, volí se optimální rychlost pro jízdu po rovince. Pokud je odchylka malá, volí maximální rychlost takovou, aby dokázal včas zareagovat a pokud možno nevyjel model z trati a pokud je odchylka velká, značí to zatáčku a vozidlo se citelně zpomalí, aby bezpečně projelo.

3.6.8 Podmínka cyklu

Hlavní smyčka (cyklus) je podmíněn dvěma stavy, mezi kterými je logický součin (čili oba musí být pravdivé k vykonání cyklu). První jsme si již uváděli výše v kapitole 3.6.2, druhá podmínka je nenulovost proměnné **int activated**, která slouží jako typ boolean. Jakmile vozítko projede například startovní čárou, tak se activated nastaví na 0 a další cykly smyčky se již neprovedou.

3.6.9 Dokončení smyčky

V cyklu se volá funkce **void sequence()**, která spustí sekvenčně všechny kroky algoritmu. Pokud během vykonávání sekvence a smyčky samotné přijde výjimka v podobě nastavení počtu cyklů či aktivační proměnné (kapitoly 3.6.8 a 3.6.2) na hodnotu 0, sekvence se dokončí a pak se vyskočí mimo smyčku. Vozítko je na trati příliš dlouho nebo došlo ke splnění projetí alespoň pěti kol (toto číslo je statistickým kompromisem mezi úspěšném projetí trati a vhodnou délkou měření). K závěru programu se zablokují motory.

3.7 Chyby a slepé cesty

Každá práce a proces získávání zkušeností stojí nejen čas a vynaložení snahy, ale i trochu toho trápení. Při práci tohoto charakteru se zde objevilo několik řešitelných i neřešitelných problémů. V krátkém pojednání v následujících podkapitolách budou přiblíženy.

3.7.1 Případy „Zdržení“

Chybné měření kamery

Kamera snímá 128 hodnot s 8bitovou hloubkou. Avšak prvních a posledních cca 9 hodnot (přesněji 10 v přední části pole a 8 v zadní) jsou nepoužitelné, neboť dochází ke špatnému vyhodnocení a často se pak tyto úseky vyhodnocovaly jako středy čáry, která tam ovšem být nemohla. Řešení omezilo možnosti kitu, ale bylo nevyhnutelné. Při vyhodnocování se použilo pouze hodnoty, které byly mimo špatné hodnocení. Na okamžik to vypadalo, že se vozítko bude muset omezit v natáčení, ale stačilo omezit jen počet čtených hodnot v poli na ty, které nebyly ovlivněny chybným měřením.

Ztratila se tím trocha citlivosti, ale nedošlo k výměně kamery ani porušení pravidel pro soutěž popsané v kapitole 1.1.2, což bylo motivací i cílem práce.

Dalším problémem s kamerou bylo měření kamery samotné. Bylo potřeba ji předsunout takovým způsobem, aby zaznamenala změnu mezi čárou a podkladem a zároveň, aby byla předsunuta natolik, aby stihl model zareagovat na zatáčku a nevyjel z trati. Optimálním řešením byla vzdálenost 7–9 cm, kdy chybovost měření byla nejnižší.

Zajímavým se také jevil fakt, že pokud autíčko vyjede z dráhy, chyba na koncích pole (indexy cca mezi 0–14 a 110–128), se prohlubuje dále do středu pole. Pokud autíčko vyjede z levé strany, chyba na začátku pole se zmenší a na konci prohloubí. Pokud z pravé strany, chyba se naopak prohloubí na začátku pole a na konci zmírní.

Vývojové prostředí

Z nějakého důvodu přestalo vývojové prostředí nahrávat algoritmus do FLASH paměti, až vzniklo podezření, že je poškozená. Algoritmus dokonce nešel nahrát

ani do RAM paměti. Ani po manuálních odpojeních či restartech nešla zprovoznit komunikace mezi vývojovým kitem a prostředím. Naštěstí po zdlouhavé přeinstalaci se vše vrátilo do původního stavu a IDE reagovalo podle požadavků uživatele.

Zaokrouhlení nízké hodnoty na nulu

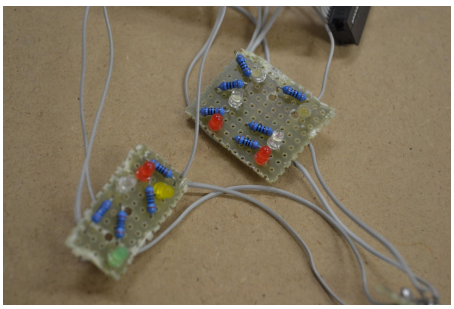
Při hledání čáry dochází k nastavení indexu středu čáry (algoritmus popsaný v kapitole 3.6.4), při němž se přepočítávalo 110 hodnot na 500 mezistupňů natočení kol. Chyba spočívala v dělení, kdy se nízké číslo zaokrouhlilo na nulu a výstupem pak bylo buď hodnota 0 či 500, takže se kola natočila úplně vlevo či vpravo. Nakonec prostým prohození dělení a nasobení se chyba ošetřila.

- Původní vzorec pro výpočet indexu:
  $line_{index} = center \times \frac{1}{110} \times 128$
- Aktuální výpočet indexu:
  $line_{index} = center \times 128 \times \frac{1}{110}$

3.7.2 Slepé cesty

Vlastní desky s LED diodami

Při řešení se realizaovaly dvě destičky s rezistory a LED diodami, které by signalizovaly důležité informace (přejetí startu, stupeň rychlosti apod.) a sloužily tak jako zpětná vazba. Avšak většina funkcí, kterou měly LED diody tvořit, zanikly s odladěním algoritmu v Terminalu. Proto na jejich použití v praxi nikdy nedošlo.

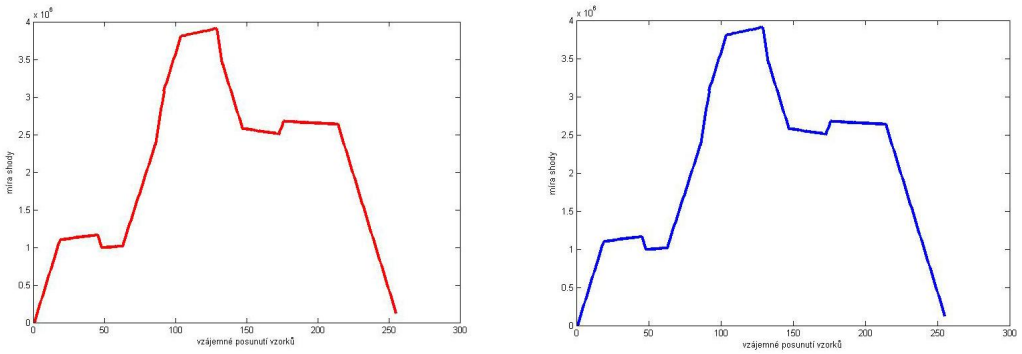


Obrázek 3.9: Vytvořené destičky s LED diodami pro vizuální zpětnou vazbu

Korelace

Korelace je v oblasti signálů skvělým nástrojem, avšak v tomto případě je skoro nepoužitelná a bylo upuštěno od tohoto způsobu hledání čáry, a zpět se navrátilo k hledání čáry pomocí prahování. Odsimulování funkce došlo jak v NetBeans, tak i v Matlabu, kdy se porovnávaly grafy z výstupu vestavěné funkce a vzniklé při naprogramování v NetBeans. Grafy (v ilustraci 3.10) se schodovaly a vypadalo to,

že korelace by mohla být velice užitečným nástrojem, už bylo za potřebí jen získat vzor. Ten byl naměřen při spuštení vozidla na dráze.



(a) Graf korelace signálu, pro odklon od středu čáry (funkce v Matlabu)

(b) Graf korelace signálu, pro odklon od středu čáry (naprogramovaná funkce)

Obrázek 3.10: Schodné grafy výstupu obou korelací

Když byl vzor načten do paměti, vozítko se snažilo autokorelací najít čáru a natočit kola úplně stejně, jak by to bývalo udělalo u hledání prahem. Avšak tento způsob měl být daleko méně závislý na osvětlení trati. Funkci samotné korelace i té části, co vyhodnocuje odchylku od středu, se podařilo naprogramovat. A vozítko díky 32bitovému procesoru skvěle s malým zdržením dokázalo korelaci spočítat, avšak ochylka byla jen zlomkem toho, co ve skutečnosti odklon od středu opravdu byl. Pravděpodobně chyba byla v samotném měření kamery, kdy si pokaždé nasnímala „nevhodný“ vzor a tato chyba se projevila na nedokonalém natočení. Od této metody se tedy upustilo a navrátilo se k hledání čáry prahem.

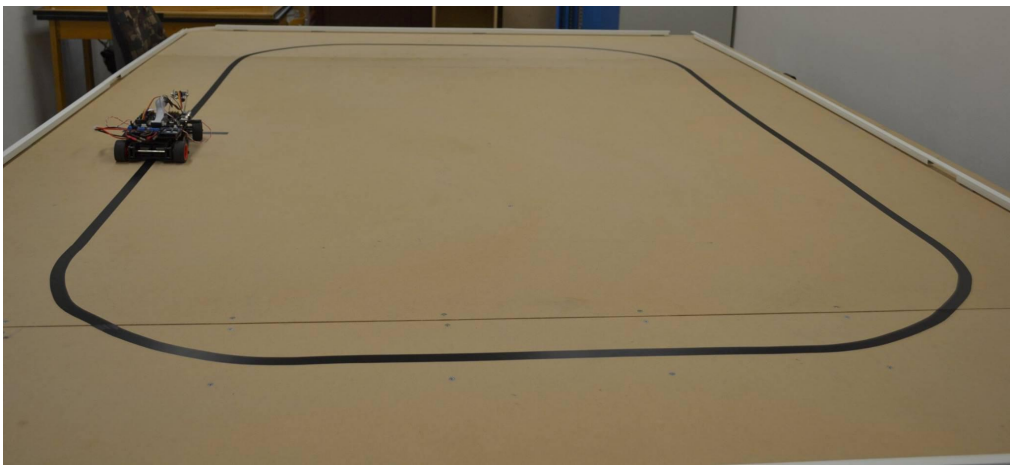
3.8 Testování na trati

Během vývoje se algoritmus nahrával do vozítka a to se pouštělo na předpřipravené trati.

Algoritmus se testoval a upravoval podle chování modelu na trati.

- Z počátku se autíčko chovalo nepředvídatelně a jen se točilo několik vteřin dokola. Takto se přišlo na chybu měření kamery (viz kapitola v 3.7.1).
- Po odladění zase mělo tendenci prudce uhýbat nebo nereagovat na přicházející zatáčku. Zde se upustilo od korelace (zdůvodnění v kapitole 3.7.2) a přešlo na prahovací funkci.
- Pak natáčelo kola pouze do extrémů a nenatáčelo kola do jiných směrů. Zde byla objevena chyba se zaokrouhlením (řešení této chyby v kapitole 3.7.1).
- V poslední fázi vozítko projíždělo trať obstojně a opatrně, avšak občas při vyjetí se nemohlo vrátit. Zde se při ladění algoritmu přišlo na ono posunutí

chybovosti do dalších prvků pole při vyjetí z trati v levém i pravém směru (kapitola 3.7.1).



Obrázek 3.11: Testovací trať

Samotné testování na trati pomohlo odladit vzniklý algoritmus do nynější podoby, kdy vozítko projede testovací trať do jedné minuty s občasnou kolizí nebo do půl minuty bez kolize. Tento problém však způsoben chybným měřením kamery, neboť někdy trať projede 6× bez zaváhání a jindy opustí trať ihned po startu. Kamera se mohla poškodit při občasné kolizi.

Závěr

Během dvou semestrů jsem se seznámil s konstrukcí i hardwarem vývojového kitu TRK. Zprovoznil jsem hardware, provedl pár vlastních úprav a začal s návrhem algoritmu, který jsem pak odladěný vyzkoušel na předpřipravené testovací dráze, kde vozítko úspěšným zajetím trati v poměrně slušném čase. I když vozítko občas vyjelo z trati z důvodu chybného měření kamerou, pruměrně jedno kolo (dlouhé 6587 mm) ujelo za 24 sekund a tím završilo i poslední bod zadání práce.

Při realizaci jsem se potkal s některými úskalími, které mi na několik okamžiků zabránily práci splnit, jak včas, tak úplně. Vždy jsem ale s odstupem času a chladnou hlavou problémy vyřešil nebo se vydal jinou cestou. A tím tak problém obešel či eliminoval úplně.

Práce mi dala mnoho hodnotných zkušeností, které bych v tomto oboru chtěl po vystudování uplatnit. Testování vozidla byla seriózní i zábavná záležitost zároveň. A to je plně motivující proč vynakládat úsilí.

Soutěže EMEA jsem se neúčastnil, ale podmínky pro účast bych splnil, bylo by však nutné vyměnit kameru a provést další testy.

Literatura

[1] RYBIČKA, Jiří. *LaTeXpro začátečníky.* Konvoj: Brno, 2003. ISBN 80-7302-049-1.

[2] ĎAĎO, Stanislav, KREIDL, Marcel. *Senzory a měřící obvody* Vyd. 2. České vysoké učení technické, 1999. ISBN 80-0102-057-6.

[3] PLÍVA, Z., J. DRÁBKOVÁ, J. KOPRNICKÝ a L. PETRŽÍLKA. *Metodika zpracování bakalářských a diplomových prací.* 2. upravené vydání. Liberec: Technická univerzita, FM 2014, ISBN 978-80-7494-049-1. Dostupné z: `http://www.fm.tul.cz/files/jak_psat_DP.pdf`

[4] Návody na citování. *Citace.com* [online]. Verze 2.0. Citace.com, 2004–2012. [Cit. 26.11.2012]. Dostupné z: `http://www.citace.com/odkazy.php`

[5] Pravidla soutěže EMEA Freescale cup *Freescale Cup 2014* [online]. Aktualizováno 7.6.2013 [cit. 2014-05-12] Dostupné z: `https://community.freescale.com/docs/DOC-94950`

[6] Datasheet pro mikrokontrolér *MPC5604B/C Microcontroller Data Sheet* [online]. [cit. 2014-05-12]. Dostupné z: `http://cache.freescale.com/files/32bit/doc/data_sheet/MPC5604BC.pdf`

[7] Uživatelský manuál k desce TRK-MPC5604B *TRK-MPC5604B EVB User Manual* [online]. [cit. 2014-05-12]. Dostupné z: `http://cache.freescale.com/files/microcontrollers/doc/user_guide/TRKMPC5604BEVBUM.pdf`

[8] Universal Serial Bus. *In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 13. 4. 2014 v 16:20 [cit. 2014-05-12]. Dostupné z: `http://cs.wikipedia.org/wiki/Universal_Serial_Bus`

[9] Korelace. *In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 7. 5. 2014 v 07:14 [cit. 2014-05-13]. Dostupné z: `http://cs.wikipedia.org/wiki/Korelace`

[10] Užité ikony k tvorbě ilustrace 2.1 *Icon Archive* [online]. [cit. 2014-05-12]. Dostupné z: `http://www.iconarchive.com`

[11] Codewarrior v.2.10 SE *CodeWarrior Development Studio for MPC55xx/MPC56xx (Classic IDE)* [online]. [cit. 2014-05-12]. Dostupné

z: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=CW-MPC55XX_56XX

[12] Princip řádkové kamery společnosti Fresscale Semiconductors *Line scan camera use* [online]. [cit. 2014-05-13]. Dostupné z: https://community.freescale.com/docs/DOC-1030

[13] Návod na zapojení obou desek plošných spojů *Wire diagram for the TRK-MPC5604B* [online]. [cit. 2014-05-13]. Dostupné z: https://community.freescale.com/docs/DOC-1019

[14] Demoprogram *Codewarrior Sample project for MPC55xxB/C and MPC56xxB/C* [online]. [cit. 2014-01-20]. Dostupné z: https://community.freescale.com/docs/DOC-1019

[15] Schéma řídicí desky *TRK-MPC5604B Schematics* [online]. [cit. 2014-01-28]. Dostupné z: http://www.freescale.com/files/microcontrollers/hardware_tools/schematics/TRKMPC5604BSCH.pdf

A Obsah přiloženého CD

Přiložené CD obsahuje:

1. Algoritmus pro řízení vozítka – CodeWarrior projekt, komprimovaný (algBP.zip).
2. Demo program – Codewarrior projekt, komprimovaný (Sample.zip).
3. Manuál k deskám plošných spojů – PDF dokument (TRKMPC5604BEVBUM.pdf).
4. Datasheet k MCU – PDF dokument (MPC5604BC.pdf).
5. Pravidla k soutěži – PDF dokument (EMEA Freescale Cup 2014 Introduction.pdf).
6. Schéma řídicí desky – PDF dokument (TRKMPC5604BSCH.pdf).
7. Zadání bakalářské práce – PDF dokument (zadani_BP_Petr_Vosoust.pdf).
8. Bakalářská práce – PDF dokument (BP.pdf).
9. Všechny užité ilustrace v bakalářské práci – složka s PDF dokumenty (Ilustrace).
10. TEX-soubory – složka se soubory pro TEX-dokument LATEX.